IoT-Based Car's Parking Monitoring System

Albertus Ega Dwiputra[1], *Handry* Khoswanto[1], *Raymond* Sutjiadi[2], and *Resmana* Lim[1,*]

[1]Electrical Engineering Department, Faculty of Industrial Technology, Petra Christian University, Jl. Siwalankerto 121-131, Surabaya, 60234, Indonesia
[2]Informatics Department, Faculty of Information Technology, Institut Informatika Indonesia, Jl. Pattimura 3, Surabaya, 60189, Indonesia

Abstract. Internet-of-things-based technologies have advanced so much and helped public necessities. The use of IoT at a parking lot will help vehicle users to know the availability of a parking location through smartphones. This IoT-based parking system is created by using controllers, sensors, servers and cloud. Controllers and sensors will be placed on the ceiling of each parking slots to detect the presence of a car. Server collect the results of the sensors and store them in Cloud. System test is conducted by installing three sensor circuits and server in a parking lot. The tests consist of measuring time that required for data transmission and the rate of success of data transmission from the parking lot to the Cloud. Based on above tests, it is observed that the sensor circuit and Radio Frequency Identification are able to transmit the parking lot data without error. This system require maximum 1 min to update parking lot data. The process of obtaining data until the data being stored in Cloud takes 12 s and the process of acquiring parking condition data from Cloud to smartphone takes 30 s. The accuracy level of parking lot data transfer is 100 %.

Key words: Internet of things, IoT cloud, parking lots, smart parking system.

1 Introduction

Along side with the development of information technology, the Internet of Things (IoT) become more popular at this time. IoT is a concept to connect various devices (smart house, car, mobile phone and household device) and enable communication between them [1]. The IoT device is equipped with electronic components, sensors, actuators and network connectivity [2]. With these components, IoT device can send data or even be controlled remotely by utilizing the internet network infrastructure.

When the concept of IoT is applied in urban life, a new concept of Smart City is introduced. Although there is no clear definition of Smart City yet, the exact goal of Smart City is to create services to facilitate access to public facilities and improve their quality [3]. This aim can be realized by spreading the use of IoT technology on the infrastructure of public facilities in urban (Urban IoT) [4]. One example of an important public facility in urban areas is public parking for four-wheeled vehicles.

There are several technologies that have been created for parking systems in Indonesia, and some of them have been used in shopping areas. For example, the concept of a smart parking lot by adding sensors, actuators, and microcontrollers. Users only need to put the vehicle at a certain location, then the microcontroller will instruct the actuator to drive the vehicle to an empty parking location [5]. This concept is very interesting, but it needs a considerable cost because it takes a total physical renovation if anyone wants to apply it on an existing public parking lot. There is also a concept of another smart parking lot using sensors, microcontrollers, and Light-Emitting Diode (LED) displays [6]. The sensor will read the amount of vacant parking space and send the data to the microcontroller. Then the microcontroller will display the amount of vacant parking spot and display it to the LED display [7].

In this part of the IoT concept can be used to improve the accessibility of parking lot for the community [8]. People do not have to go to the location to find out how much parking space is available and where the parking lot is empty.

[*]Corresponding author: resmana@petra.ac.id

The use of mobile application connected with cloud can tell where the parking spaces are empty. The use of Radio Frequency Identification (RFID) can be added to the parking lot to increase the ease of data collection that can be used for the analysis process [9].

Therefore it is necessary to create a smart parking system that can connect with cloud. This parking system can help many vehicle users to find parking space more easily. Data obtained from this parking system can be sent to mobile phones or other applications. Communities can monitor vacant parking lots remotely and in real-time.

2 System design

The developed parking system is a system that is able to calculate and detect the presence of cars in the parking area. The data obtained in the parking area will be sent to the cloud database. The data will be used by Android application to inform the users where parking space they should go to. Figure 1 shows the mechanism of the parking system.

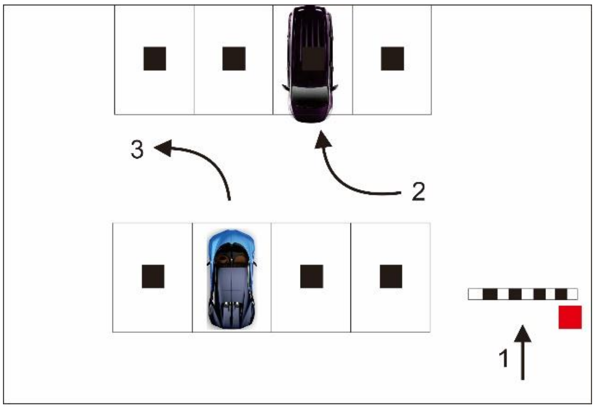

Fig. 1. Mechanism of parking system.

There are three main processes that become the core of this parking system design:
(i) Payment process at the entrance gate.
(ii) Signing in to the parking slot.
(iii) Processing out of the parking slot.

The first process is payment at the entrance, where at the entrance gate will be placed RFID reader and controller with Liquid Crystal Display (LCD). Car driver needs to have RFID card that has a function as e-money. The car driver only needs to tap the card on the scanner and data in the card will be sent to the local server and cloud then the gate will open. In this chapter, only a series of data readings and transmissions will be created and does not involve the development of a parking gate bar. However, if necessary, the microcontroller and RFID reader circuit can be connected to the parking gate bar because the relays are provided in the circuit.

The second process is the process of getting into the parking slot. In each of the parking slots will be installed car detection sensors and additional sensors on the parking marker to keep the car park tidiness. Each slot has a unique address for the parking lot mapping process. If a car is detected at a specific slot address, then the system will send the latest data to the cloud for updating. The third process is the process of getting out of the parking slot. This process is almost the same as the second process. If the sensor detects the car exits a certain parking slot, then the system will update the existing data in the cloud according to the slot address under changing conditions.

3 Circuit design

This system consists of two main circuits, i.e. sensor circuit and RFID sensor.

3.1 Sensor circuit

The controllers used in this system are Arduino Pro Mini and the modules are ESP8266, two HC-SR04, 16 × 2 LCD and a SHARP GP2Y0A710K0F proximity sensor. Other supporting components are 2N2904 transistor, 3.3 volt 1N4728A zener diode, LED, buzzer and resistor. Input for microcontroller comes from three sensors: two HC-SR04 ultrasonic and one GP2Y0A710K0F proximity sensor. Both sensors are used to detect the presence of the car. Proximity sensor is used to detect the distance of the existing parking limit on the side. If there is a car occupy more than its own parking area then the output value of the sensor will be processed by the controller, then the buzzer will be turned on to let the driver knows that the car's position is improper.

Two LEDs are used only to provide information visually about the presence of the car. If there is no car in parking slot, then one LED will light up, and if the parking slot occupied, another LED will light up. LCD is also used as a visual information that reads the three sensors. LCD does not need to be always installed, because the position of the circuit located in the ceiling of the parking area. ESP8266 WiFi Module serves as a data sender medium from the microcontroller to the local server.

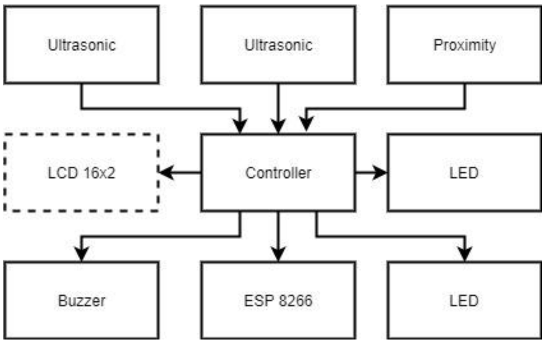

Fig. 2. Sensor circuit scheme.

3.2 RFID circuit

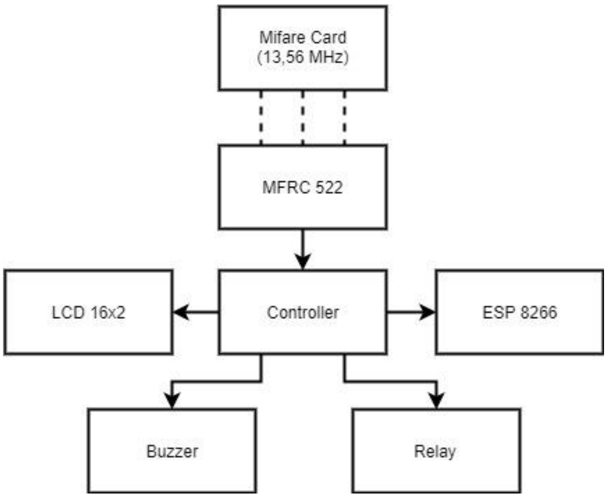

Fig. 3. RFID circuit scheme.

Figure 3 is a circuit diagram used for RFID card processing. In this system, the RFID reader is MFRC522 and the RFID card that can be used is the card which suits the 13.56 MHz protocol. ESP-8266 is used as a communication medium between local and Arduino servers.

4 Software design

Software design is divided into three parts: software for sensor circuit, software for RFID circuit and software for server.

4.1 Sensor circuit software

Figure 4 below illustrates how the system starts by initializing variables, I/O ports, and libraries used. By default, the sensor circuit will assume the parking slot is empty. Once all are initialized, the Arduino will calibrate the sensor by sampling the sensor readout data hundreds times. The average reading result will be a normal distance when no car is detected.

When calibration is complete, the Arduino will enter the network using ESP8266. Once connected with the network, the Arduino is ready to perform sensor readings. First, Arduino will read the existence of the car using both ultrasonic sensors. If both sensors detect a distance difference of approximately 1.3 m, then it means there is a car in that blocks and the presence status of the car will be changed to one. But if there is no car, the presence status of the car will be changed to zero.

If parking status has been obtained, the Arduino will check the status of the previous car with the current car status. If previously there was no car and now there is a car, or otherwise, then Arduino will make a connection to the local server and send the data of the parking slot. However if current and previous conditions are the same, Arduino will not establish a connection with the server. The next process is checking the parking tidiness. The circuit will do the distance reading on the proximity sensor to detect parking that is untidy. If the car is detected then the buzzer will be sounded, otherwise the buzzer will be turned off. After that, the program will perform a delay for one second then do the repetition checking the existence of the car.

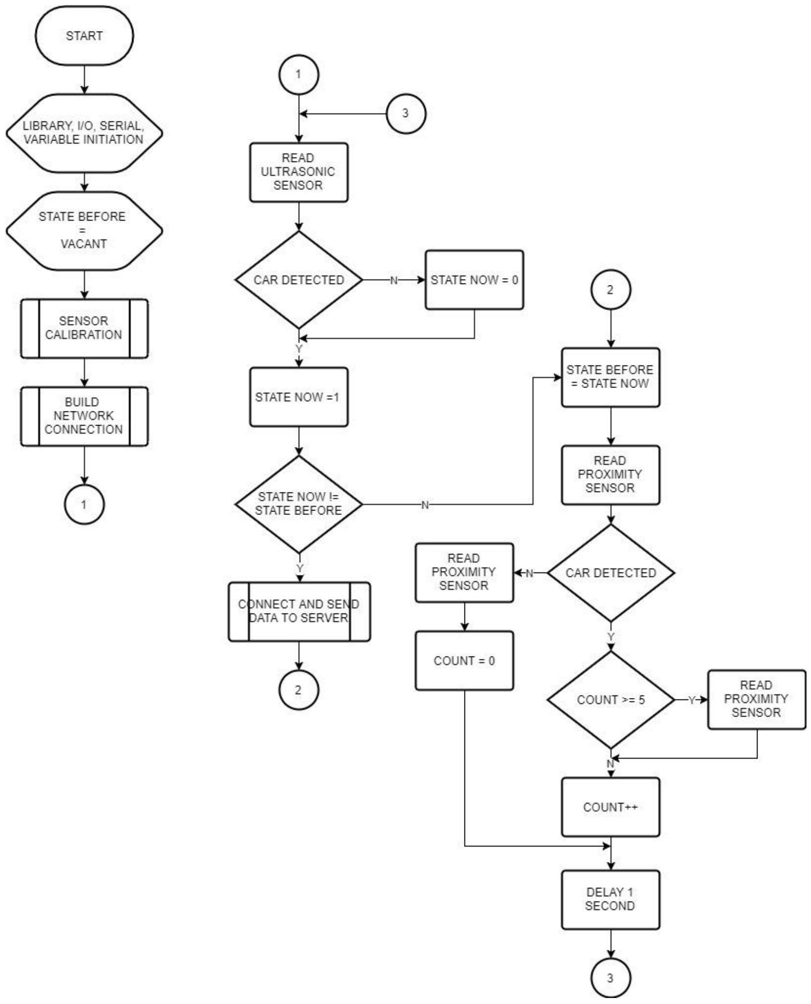

Fig. 4. Sensor flowchart.

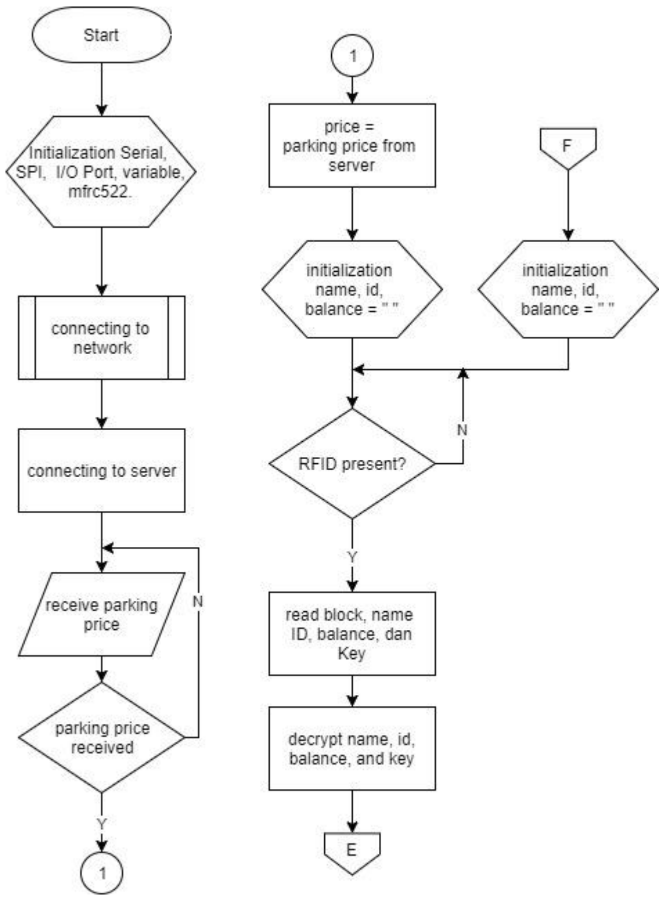
Start
Initialization Serial, SPI, I/O Port, variable, mfrc522.
connecting to network
connecting to server
receive parking price
parking price received
N
Y
1
1
price = parking price from server
initialization name, id, balance = " "
F
initialization name, id, balance = " "
RFID present?
N
Y
read block, name ID, balance, dan Key
decrypt name, id, balance, and key
E

Fig. 5. RFID flowchart (part 1).

As shown in Figure 5, at the beginning, the program will initialize variables and libraries. After that, the circuit will be connected to WiFi network. In contrast to the sensor circuit design software, the circuit will be directly connected to the server to request parking fee from the server. If the fee is accepted, the process will resume. But if not yet, the request will be resent. If the parking fee has been successfully received, the data will be stored in a variable.

Once the connection is established and the fee is received, the program will initialize the variables to store the existing data on the RFID card. The variable will be emptied and the program will start the RFID card readings if any card is detected. If the card is detected then the data in the card will be read, decrypted and stored into the variables that have been provided. The received variables will be converted according to the needs because some data needs an integer or double and string type, while the reading data is a char array.

Continue to Figure 6 below, after the data is saved, card checking will be done to check whether the balance is sufficient or not. If not pass the existing requirements, then the card will be rejected and the circuit will be ready again to read the next card. But if the card complies with the existing requirements, then the connection to the server will be reconnected and data will be sent. After the data delivery is complete, the balance will be deducted, encrypted, and re-written to the card. Then the circuit will be prepared to read the next card.

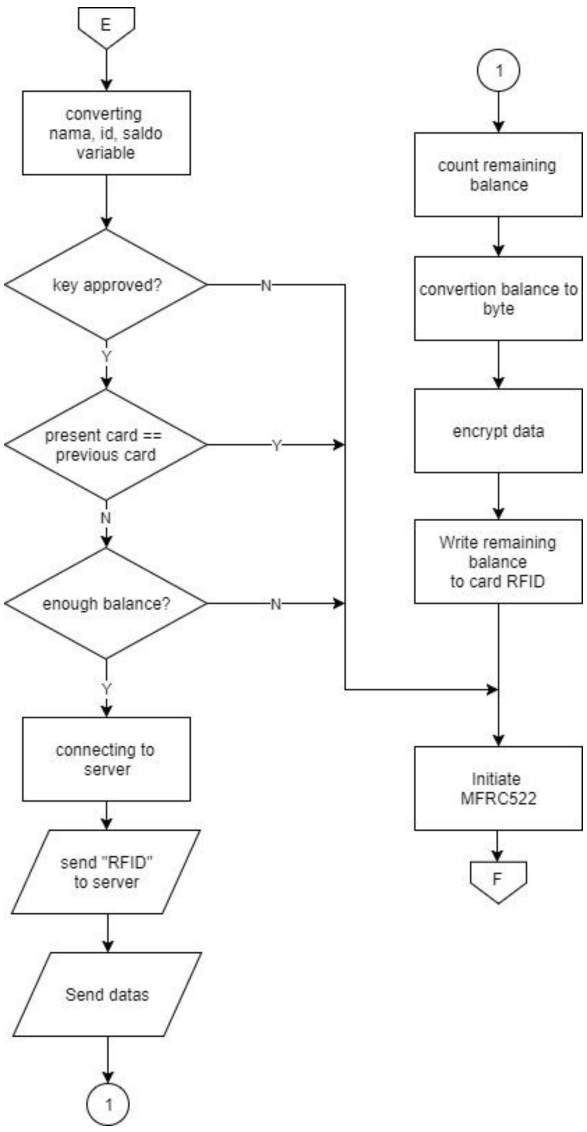

Fig. 6. RFID flowchart (part 2).

4.3 Local server software

The languages used on the server are Java and Python languages. The server works as a temporary receiver of data from sensors installed in the parking area. The data will be sent to cloud with the help of internet network. Figure 6 is the server program logic.

The server is designed using multithread Transmission Control Protocol (TCP) so it can handle more than one connection [10]. The first thing to do is to initialize the variables, sockets and classes needed for receiving and sending data streams. After that the server will take the fee of parking space stored in the cloud. Once the fee is stored in the server, a socket for a TCP connection will be established. If there is a connection request, then a new thread will be formed and run the server function.

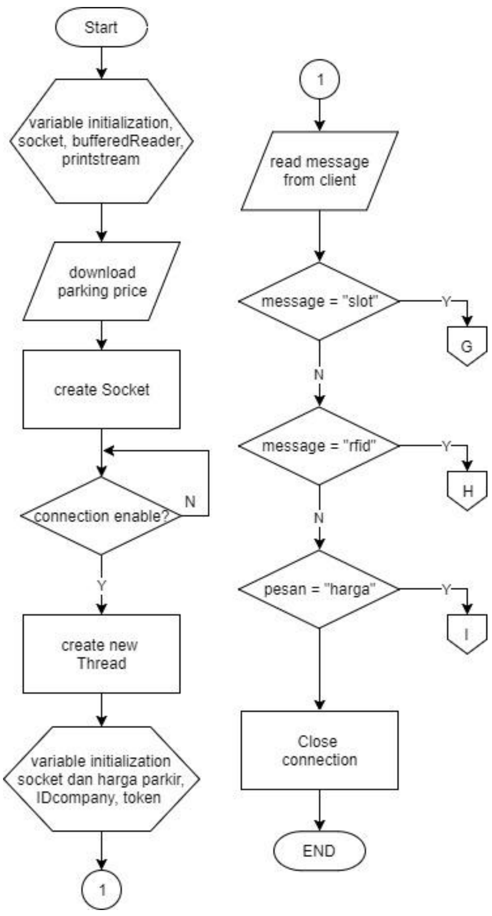

Fig. 7. Server flowchart part 1.

In the new thread, socket variables will be initialized, the parking fee, the company ID, and the token used. After that, the server will wait for data confirmation from the client connected to the server. There are three types of client requests to the server i.e. slot, RFID and parking fee. If the server can not identify the data sent, the connection will be disconnected by the server.

(i) Slot: If the request slot provided by the client, the server will again wait for data to be sent by the client. The data sent will be separated first and stored into several variables. The data received is the address and status of the parking slot. After that the server will take the data address of the variable ID in the database. Once the data is found, variable ID address of the database will be used to send the latest data to the server.

(ii) RFID: If the RFID request provided by the client, the server will wait for data to be sent by the client. The data sent will be separated first and stored into several variables. The data received are name, ID card and balance. After that, the server will retrieve the ID variable stored in the database. The existing variable ID will be used to send the RFID card ID to the cloud according to the address.

(iii) Parking fee: If the fee request provided by the client, the server will wait for one and a half seconds then the server will send the fee data back to the client. A one-and-a-half-second delay is done to allow the RFID circuit time to readily read the data serially.

After the process of one of the above three processes is completed then the server will disconnect and the program is completed.

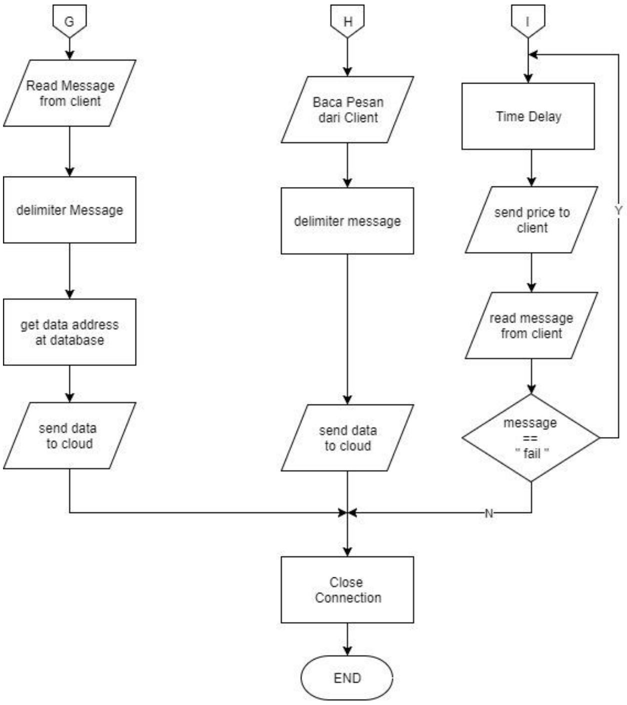

Fig. 8. Server flowchart part 2.

In the new thread, socket variables will be initialized, the parking fee, the company ID, and the token used. After that, the server will wait for data confirmation from the client connected to the server. There are three types of client requests to the server i.e. slot, RFID, and parking fee. If the server can not identify the data sent, the connection will be disconnected by the server.

(i) Slot: If the request slot provided by the client, the server will again wait for data to be sent by the client. The data sent will be separated first and stored into several variables. The data received is the address and status of the parking slot. After that the server will take the data address of the variable ID in the database. Once the data is found, variable ID address of the database will be used to send the latest data to the server.

(ii) RFID: If the RFID request provided by the client, the server will wait for data to be sent by the client. The data sent will be separated first and stored into several variables. The data received are name, ID card and balance. After that, the server will retrieve the ID variable stored in the database. The existing variable ID will be used to send the RFID card ID to the cloud according to the address.

(iii) Parking fee: If the fee request provided by the client, the server will wait for one and a half seconds then the server will send the fee data back to the client. A one-and-a-half-second delay is done to allow the RFID circuit time to readily read the data serially.

After the process of one of the above three processes is completed then the server will disconnect and the program is completed.

5 System testing

This system testing is conducted in two main circuits, i.e. sensor circuit and RFID sensor.

5.1 Sensor circuit testing

The test of the circuit data communication of the whole circuit to and from cloud is done by detecting the existence of the car 10 times with the assumption of 150 cm car height and floor to ceiling distance of 350 cm. The measuring point of this test is the success of the circuit in sending data, the length of time required, and the validity of the data. Calculation of respond time on the Arduino and server is done by the program by recording the difference between the start and end time of the program. The calculation of respond time on cloud is done by calculating the difference between server sent time and Ubidots received time. Based on the testing result as shown in Table 1 below, The sensor circuit has 100 % success rate for sending data to cloud with an average total required time of 6.4004 s for each data. Data errors were not found during the test.

Table 1. Sensor circuit testing result.

Test	1010001	Response Time			Total Time (s)	Data Error
		Arduino (ms)	Server (ms)	Ubi dots (s)		
1	Pass	237	228	4	4.465	0
2	Pass	249	315	6	6.564	0
3	Pass	312	226	6	6.538	0
4	Pass	369	235	6	6.604	0
5	Pass	253	234	6	6.487	0
6	Pass	245	209	6	6.454	0
7	Pass	232	210	6	6.442	0
8	Pass	274	229	6	6.503	0
9	Pass	233	225	6	6.458	0
10	Pass	268	221	7	7.489	0
Avg	100 %	267.2	233.2	5.9	6.400 4	0 %

5.2 RFID circuit testing

The method used to test RFID circuit communications to cloud is by tapping RFID cards ten times. In the program, there are already methods of calculating the respond time such as testing the sensor circuit. The measured points are the success of the circuit sending the data, the time it takes to send from the circuit to cloud, and the validity of the data sent with the received.

Table 2. RFID circuit testing result.

Test	RFID 1	Response Time			Total Time (s)	Data Error
		Arduino (ms)	Server (ms)	Ubi dots (s)		
1	Pass	1 396	751	7	9.147	0
2	Pass	925	362	8	9.287	0
3	Pass	1 406	776	7	9.182	0
4	Pass	1 303	550	7	8.853	0
5	Pass	2 156	500	2	4.656	0
6	Pass	1 488	603	7	9.091	0
7	Pass	1 714	622	8	10.333 6	0
8	Pass	1 214	161 4	6	8.828	0
9	Pass	1 062	121 1	3	5.273	0
10	Pass	1 302	546	6	7.848	0
Avg	100 %	1 396.6	753.5	6.1	8.250 1	0 %

RFID circuits have 100 % success rate in sending data to Ubidots cloud with the percentage of sent data error is 0 %. The average time it takes to send data from scratch when the card is detected until the data received by Ubidots is 8.2501 s. The process in Arduino takes an average of 1 396.6 ms and 753.5 ms is on the server. The longest time spent when sending data from the server to Ubidots is 6.1 s.

5.3 Car detection

Testing the accuracy of sensor readings conducted by car detection test with the assumption that the car has a height of 150 cm. Testing is done as much as ten times for each circuit of sensors by shifting obstacles. If there is a parking car, then the obstacles will be placed between the sensor and the floor. If the condition of the parking slot is empty, then the barrier will be removed so that the sensor will immediately detect the floor.

Table 4. Car detection testing with occupied parking slot condition

Car Present (Car Height 150 cm) - (Ceiling distance 350 cm)						
Test	**1010001**	**1010002**	**1010003**	**1010004**	**1010005**	**1010006**
1	Detected	Detected	Detected	Detected	Detected	Detected
2	Detected	Detected	Detected	Detected	Detected	Detected
3	Detected	Detected	Detected	Detected	Detected	Detected
4	Detected	Detected	Detected	Detected	Detected	Detected
5	Detected	Detected	Detected	Detected	Detected	Detected
6	Detected	Detected	Detected	Detected	Detected	Detected
7	Detected	Detected	Detected	Detected	Detected	Detected
8	Detected	Detected	Detected	Detected	Detected	Detected
9	Detected	Detected	Detected	Detected	Detected	Detected
10	Detected	Detected	Detected	Detected	Detected	Detected

Table 5. Car detection testing with unoccupied parking slot condition.

Car Not Present - (Ceiling distance 350 cm)						
Test	**1010001**	**1010002**	**1010003**	**1010004**	**1010005**	**1010006**
1	Empty	Empty	Empty	Empty	Empty	Empty
2	Empty	Empty	Empty	Empty	Empty	Empty
3	Empty	Empty	Empty	Empty	Empty	Empty
4	Empty	Empty	Empty	Empty	Empty	Empty
5	Empty	Empty	Empty	Empty	Empty	Empty
6	Empty	Empty	Empty	Empty	Empty	Empty
7	Empty	Empty	Empty	Empty	Empty	Empty
8	Empty	Empty	Empty	Empty	Empty	Empty
9	Empty	Empty	Empty	Empty	Empty	Empty
10	Empty	Empty	Empty	Empty	Empty	Empty

There are six circuits that are used for the testing process. From the two tables of sensor testing, it can be concluded that the whole circuits of managed sensors is able to detect the presence of the car accurately.

6 Conclusion

The development of a prototype of the internet-based smart parking system works well in detecting, processing parking data, and sending parking data to the cloud. Here are the key points that can be deduced from testing data are sensor and RFID circuits can send data to Ubidots IoT cloud without any data error under twelve seconds. The second is the RFID circuit is capable of reading and writing data back to the card with no data error under two seconds. The third is sensor, RFID, and server circuits can process data in less than two seconds. The last is data delivery to cloud takes longer than sending data from circuit to server, with maximum time is 20 s.

References

1. G. Lobaccaro, S. Carlucci, E. Löfström. Energies, **9**(5):1–33 (2016). http://www.mdpi.com/1996-1073/9/5/348.
2. P. Sethi, S.R. Sarangi. Journal of Electrical and Computer Engineering, **2017**(9324035):1–25 (2017). https://www.hindawi.com/journals/jece/2017/9324035/
3. Economic and Social Council, United Nations. *Smart cities and infrastructure: Report of the secretary-general*. Geneva: Commission on Science and Technology for Development (2016). http://unctad.org/meetings/en/SessionalDocuments/ecn162016d2_en.pdf
4. A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi. IEEE IoT Journal, **1**,1:22–32 (2014). http://doi.org/10.1109/JIOT.2014.2306328
5. I. Winarsih, R. Mahendra. JETri, **8**,2:21–36 (2009). [in Bahasa Indonesia]. http://download.portalgaruda.org/article.php?article=253389&val=6824&title=SISTEM%20PARKIR%20OTOMATIS%20MENGGUNAKAN%20RFID%20BERBASISKAN%20MIKROKONTROLER%20AT%2089S51

6. M. Priyatham M, S. Mandal S., M.S. Lakshmi, L. Ashwani, B.M. Sandhya. IJERCSE, **4**(6):360–363 (2017). http://technoarete.org/common_abstract/pdf/IJERCSE/v4/i6/Ext_03295.pdf
7. A. Anita. *Perancangan dan implementasi sistem alokasi tempat parkir berbasis mikrokontroler ATMega8535*. [Design and implementation of ATMega8535 microcontroller based parking allocation system] [Thesis]. Teknik Telekomunikasi, Fakultas Ilmu Terapan, Universitas Telkom (2011). [in Bahasa Indonesia]. http://repository.telkomuniversity.ac.id/pustaka/97105/perancangan-dan-implementasi-sistem-alokasi-tempat-parkir-berbasis-mikrokontroler-atmega8535.html
8. Z. Ji, I. Ganchev, M. O'Droma, L. Zhao, X. Zhang. Sensors **14**:2272–22393 (2014). http://www.mdpi.com/1424-8220/14/12/22372/pdf
9. P.C. Jain, K.P. Vijaygopalan. *RFID and wireless sensor networks*. Proceedings of ASCNT (Noida, India, 2010). https://pdfs.semanticscholar.org/47ae/cbb03c54d74911e11266d24289586ac72e2a.pdf
10. T.J. Andersson. *Single SCTP association with multiple streams vs. multiple TCP connections: A performance evaluation* [Thesis]. Department of Computer Science, Karlstad University (2005). https://www.cs.kau.se/cs/education/courses/davddiss/Uppsatser_2005/D2005-08.pdf